Time for Real-Time Temporal Databases?*

Krithi Ramamritham

Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{*krithi@nirvan.cs.umass.edu*}

February 28, 1996

Abstract

Time related concepts appear in both temporal databases and real-time databases. In this position paper, we show that with minor changes to the recently compiled basic concepts of temporal databases, it is possible to cover the conceptual needs of real-time databases as well. We also discuss practical implementation-oriented issues that arise when real-time databases are married to temporal databases. The motivation behind this paper is to determine whether, as we attempt to standardize temporal database concepts, it is possible to take into account the specific characteristics of real-time databases as well so as to expand the applicability of temporal databases.

1 Introduction

Temporal databases [3] are designed to capture time-varying information. They maintain the history of changes that objects in a database have undergone. Real-time databases [2], on the other hand, have been more concerned with the timely processing of information. Queries and transactions in real-time databases have time constraints, such as deadlines, associated with them and *when* a result is produced is as important as what the result is. Thus on the surface, it might appear that temporal databases and real-time databases have orthogonal goals. A *real-time temporal* database may *appear* to be one where queries and transactions – over a temporal database – have timing constraints.

In the rest of this paper, we will show that there is more to real-time temporal databases and that temporal databases already have the necessary conceptual ingredients to accommodate real-time database concepts. We will provide details of the similarities and differences between the two types of time-related databases and motivate the need to examine them synergistically.

2 Real-Time Databases

In this section we will examine real-time databases from the viewpoint of the relationship between the database and its environment, consistency requirements, and transaction and query processing considerations.

2.1 Relationship between the database and its environment

Typically, any computer-controlled real-world application can be viewed in terms of *the controlling system* and the *controlled system.* For example, in an automated factory, the controlled system is the factory floor with its robots, assembling stations, and the assembled parts, while the control-

*This material is based upon work supported by the National Science Foundation under grants IRI-9114197, CDA-8922572, and IRI - 9109210.

ling system is the computer and human interfaces that manage and coordinate the activities on the factory floor. Thus, the controlled system can be viewed as the *environment* with which the computer interacts.

The controlling system interacts with its environment based on the data available about the environment, say from various sensors, e.g. position sensors. It is imperative that the state of the environment, as perceived by the controlling system, be consistent with the actual state of the environment. Otherwise, the effects of the controlling systems' activities may be disastrous. Hence, timely monitoring of the environment as well as timely processing of the sensed information is necessary. The sensed data is processed further to derive new data. For example, the position and velocity of moving objects on a factory floor may be used to determine a new path for a robot. This derivation typically would depend on the characteristics of the objects and so some of the needed information may have to be fetched from archival storage. Based on the derived data, where the derivation may involve multiple steps, actuator commands are set. For instance, in our example, the derived path is used to determine the next command to be given to the robot. In general, the history of (interactions with) the environment are also logged in archival storage. This example shows that there is need to gather data from the environment, process the gathered information in the context of information acquired in the past, and provide *timely* response. Another aspect of this example is that it involves processing both temporal data, such as the current position of the objects, which loses its validity after a certain interval, as well as archival data, such as the object characteristics.

In addition to the timing constraints that arise from the need to continuously track the environment, timing correctness requirements in a real–time (database) system also arise because of the need to make data available to the controlling system for its decision-making activities. For example, if the computer controlling a robot does not have up-to-date information about a robot's position then it may not be able to command the robot to stop or turn on time, and the robot might collide with another object on the factory floor. Needless to say, such a mishap can result in a major catastrophe.

Temporal consistency requirements of data along with time constraints on system response are the two distinguishing features of real-time databases. We discuss these two issues next.

2.2 Temporal Consistency Requirements

In real-time databases, the need to maintain consistency between the actual state of the environment and the state as reflected by the contents of the database leads to the notion of temporal consistency [4] which has two components:

- *Absolute consistency* – between the state of the environment and its reflection in the database; this arises from the need to keep the controlling system's view of the state of the environment consistent with the actual state of the environment. Let us denote a data item in the real-time database by "$d : (value, avi, timestamp)$" where d_{value} denotes the current state of d, and $d_{timestamp}$ denotes the time when the observation relating to d was made. d_{avi} denotes d's *absolute validity interval*, i.e., the length of the time interval following $d_{timestamp}$ during which d is considered to have absolute validity.

- *Relative consistency* – among the data used to derive other data; this arises from the need to produce the sources of derived data close to each other. A set of data items used to derive a new data item form a *relative consistency set*. Each such set R is associated with a *relative validity interval* denoted by R_{rvi}.

Assume that $d \in R$. d has a correct state iff

1. d_{value} is logically consistent – satisfies all integrity constraints.

2. d is temporally consistent:

   - Absolute consistency: $(current\_time \ - \ d_{timestamp}) \ \leq \ d_{avi}$.
   - Relative consistency: $\forall d' \in R$, $\mid d_{timestamp} \ - \ d'_{timestamp} \mid \leq \ R_{rvi}$.

Some time constraints on database activities come from the need to maintain temporal consistency of data and some come from requirements imposed on system reaction time. The former typically take the form of periodicity requirements while system reaction requirements typically take the form of deadline constraints. As discussed next, the processing of transactions must take their different timing requirements into account.

2.3 Time Constraints on Processing Transactions and Queries

A majority of time constraints result from the need to maintain temporal consistency of objects. These are typically periodicity constraints. Let us assume that periodic transactions are scheduled so that each instance of a transaction is guaranteed to start at the same time, relative to the beginning of a period. Since a transaction could make an observation and write a database object any time during its execution, the worst case separation between the start time of one instance and the finish time of the subsequent instance will be $(P \ + \ (2 * e))$, where e is the execution time of a transaction, and P is the period. This interval must be no larger than the *avi* of the object. Thus, $P = (avi \ - \ (2 * e))$. In a similar manner, depending on the assumptions we make about the periodic executions, different constraints will be imposed on the period.

Some requirements produce transactions that are aperiodic. Examples include the constraints that are mandated by the system reaction requirements imposed by the physical aspects of the environment with which the system is interacting. For instance, in the case of a system controlling a robot, the characteristics of the robot dictate how soon the system must react to an abnormal condition, such as the detection of an unanticipated obstacle, on the factory floor. Whether or not a condition is abnormal will typically depend on not only the current state of the robot but also on that of the other objects on the floor and their characteristics.

A lot of work has been done in the real-time database area to process transactions and queries under time constraints. A review can be found in [2]. It should be clear by now that one of the primary reasons to satisfy the time constraints is the *active* maintenance of the temporal consistency requirements. That is, we desire to make sure that the data in the database is always valid. The system does this by attempting to update a database object before it becomes invalid. This active pursuit of validity maintenance by proper processing techniques is a crucial ingredient of real-time databases.

3 Temporal Databases and Real-Time Databases

We can now see the connections between real-time databases and temporal databases. Clearly, temporal databases provide easy means to maintain the past history of a controlled system. Also, the concept of valid-time lifespan [1] can be used to capture the loss of validity of an observation about the controlled system beyond a certain point in time. What temporal databases do not appear to have is a means to be pro-active in the maintenance of the validity of the data. Said differently, they do not *explicitly* consider the time constraints on query and transaction processing. Let us elaborate upon these in the following subsections. They compare and contrast temporal databases and real-time databases along the three dimensions considered in the previous section.

3.1 Relationship between the database and its environment

Temporal databases have been motivated by a need to maintain the time-varying characteristics of database objects. Objects are endowed with valid times and transaction times. A change in the state of a real-world object does not cause an in-place modification in its database counterpart but rather involves the addition of a new entry in the database. A database object has a lifespan that denotes its interval of validity. Objects continue to be in the database even after this interval expires. Transactions can be invoked that access data objects that were valid at some point in the past.

In contrast, real-time database objects, at least the way they have been viewed thus far, have objects that are updated in-place and, in most cases, a transaction is allowed to access only current, i.e., valid, information in the database. Also, as we saw, the controlled system is tracked constantly to keep the database current.

To summarize, temporal databases can be used to keep track of changes in the real-world, but perhaps because objects continue to be in the database even after their validity expires, there is no active involvement of the temporal database to keep up with the changes in the environment.

3.2 Capturing Temporal Consistency Requirements in Temporal Databases

To make the connections between concepts in real-time databases and temporal databases, we quote from the definitions of the relevant temporal database terms as they appear in [1].

Valid-time of a fact is defined to be "the time when the fact is true in the modeled reality". Hence valid time can be used to capture the timestamp of a database object. *Lifespan* of a database object is "the time over which it is defined". *Valid-time lifespan* "is the time when the corresponding object exists in the modeled reality". If lifespan denotes an interval, we see the ability of valid-time lifespan to model absolute consistency requirements.

Since things look promising let us continue and see if there is an analog of relative consistency requirements in temporal databases.

The notion of *temporal homogeneity* is relevant here. A temporal tuple is temporally homogeneous "if the lifespan of all attribute values within it are identical". The need for the lifespans, in our case, transaction-time lifespans, to be identical is clearly much stronger than is needed for relative consistency, given our definition in the previous section. However, if it is possible to maintain temporal homogeneity, we are guaranteed to satisfy relative consistency requirements.

At this point we would like to introduce some practical considerations. We saw earlier that a given absolute consistency requirement can be realized by sampling the corresponding real-world parameter often enough. However, ensuring relative consistency may not be that straightforward. This is because, achieving relative consistency implies that the data items that belong to a relative consistency set have to be observed at times close to each other.

Also, achieving an *rvi* along with the *avi*'s will mean that smallest of the *avi*'s of the data items belonging to the relative consistency set will prevail. Consider *temperature* and *pressure* belonging to a relative consistency set R. Suppose $temperature_{avi} = 5$, $pressure_{avi} = 10$, $R = \{temperature, pressure\}$, and $R_{rvi} = 2$. The transactions writing *temperature* and *pressure*, respectively, must always write them within 2 time units of each other. This will implicitly lower the *avi* of *pressure* to 5. One way out of this predicament is to realize that relative consistency requirements result from the need to derive data – from data produced within close proximity of each other. Thus meeting relative consistency requirements is necessary only when data is *used* to derive other data. So rather than reducing the *avi*'s, we need to ensure that an *rvi* is satisfied just when a transaction is executed to derive new data.

Given these practical considerations, clearly, temporal homogeneity is, from a practical point of view, even stronger than we thought earlier. Also, temporal homogeneity is defined [1] with respect to all attribute values of a tuple or all tuples of a relation. Thus, to cater to the needs of real-time data requirements, we need to weaken this. Specifically, we need a way by which to specify the set of attributes or tuples or relations that correspond to a relative consistency set, that is, that have to be temporally homogeneous. One possibility is to use the *select* and *project* operations to construct views and associate each relative consistency set with such a view.

3.3 Processing Transactions and Queries

As we said earlier, whereas transactions in real-time databases are associated with time constraints so as to ensure temporal consistency of the database objects, the same is not true in temporal databases. This is perhaps the biggest difference between them and has to be addressed to bring the two together.

Since, on the other hand, techniques for processing time constrained transactions and queries have been investigated extensively for real-time databases [2], they can be applied to situations where temporal databases are employed. With this, we have a way by which to denote not only when a database object is valid but also make sure that database objects are updated in such a way that a valid entry exists at the current time.

An open issue is whether the basic infrastructure being discussed currently for temporal databases must support the transaction processing mechanisms needed for time constrained transactions and queries. Since many aspects of temporal databases remain fluid, perhaps a workable strategy would be to design the infrastructure in such a way that it does not preclude the use of real-time database concepts and mechanisms.

4 Conclusions

Many real-time database transactions use past trends in the environment to determine their output. For instance, besides automated factories, in program stock trading, reactor control, and network management, one examines past behavior of the controlled system to determine subsequent responses from the controlling system. Temporal databases provide the necessary facility to maintain this historical information. They also have the notion of validity which has its counterpart in real-time databases, in the form of temporal consistency. What temporal databases do not appear to have is an active means to maintain validity with respect to the current time, but real-time databases make this their primary concern. So we see that temporal databases and real-time databases complement each other in their capabilities and a lot is to be gained by considering the two in conjunction.

References

[1] C.S. Jensen, J. Clifford, S.K. Gadia, A. Segev, R.T. Snodgrass, "A Glossary of Temporal Database Concepts", Technical Report, July, 1992.

[2] K. Ramamritham, "Real-Time Databases." *Int'l Journal of Distributed and Parallel Databases*, Vol. 1, No. 2, March 1993.

[3] R. Snodgrass and I. Ahn, "Temporal Databases", *IEEE Computer*, Vol 19, No. 9, September 1986, pp. 35-42.

[4] X. Song and J.W.S. Liu, "How Well Can Data Temporal Consistency be Maintained?" *Proceedings of the IEEE Symposium on Computer-Aided Control Systems Design*, 1992.